Take-Home Assignment:

Question #1: (Selenium-driven Checkout (Front End))

Note: Although Python is the preferred languages to use to complete this exercise, it is acceptable to use any other language that currently supports the latest Selenium-WebDriver bindings: https://www.seleniumhq.org/docs/. However, if using other languages, include comments on how to compile - if applicable - and run your test script.

Using Selenium-WebDriver to drive either Chrome or Firefox to do the following:

1. Go to https://www.saucedemo.com/

2. Login using the “standard user” and password.

3. Add the a **“third item”** to the cart

Note: In the next section, please use a module/library to generate “fake” random data for the text fields (e.g names, password) such as:

- Faker in Python: https://faker.readthedocs.io/en/master/
- Mimesis in Python: https://mimesis.readthedocs.io
- Faker in Java: https://github.com/DiUS/java-faker

If using languages other than Python/Java, look for a Faker port or an equivalent module/library that can create fake test data on demand for your language of choice.

4. Navigate to the Cart and checkout the item. Fill in the following information:

- First Name
- Last Name
- Zip/Postal Code

5. Click Continue and Finish.

6. Important step: Assert that the item has successfully been ordered/confirmed.

***Important**: **Please upload your selenium assignment to github or google drive**, save your code in a file named selenium_checkout(.py/.java) and provide a link to your work in the email. Specify in comments within the file which exact version of Python or Java was used (do not use Python 2.x).*

Question #2 (API Testing)

Using a platform of your choice (Postman, restAssured, integrated with pytest using python request library, java library, etc)

- Go to https://openweathermap.org/api
- Subscribe to a free account to receive the API key.

1. Print the current temperature of the city **Toronto** in **Celsius**
2. Print the status 200 from a successful API response.

Question #3 (Test Case Writing)

ATM Bank Machine that dispenses cash.

Create a test plan for how you would ensure it's functions and behaviours are correct. Within the test plan, come up with different test cases that you can think of and group them by similar components. **Minimum 8 test cases**.

Ex.
Component "A"
1. Feature A should behave like this
2. Feature A should behave like that

Component "B"
1. Feature B should behave like this
2. Feature B should behave like that

Don't spend time coming up with really detailed steps. A descriptive test case title and expected results are good enough; we are more interested in creativity and thoughtfulness.

Save this in a pdf and include it together with the completed question #1 in an email.